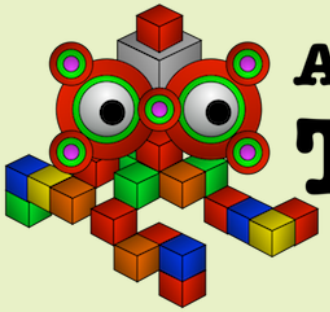
The Gremlin Graph Traversal Language



Apache TinkerPop

http://tinkerpop.incubator.apache.org

Marko A. Rodriguez and Daniel Kuppitz

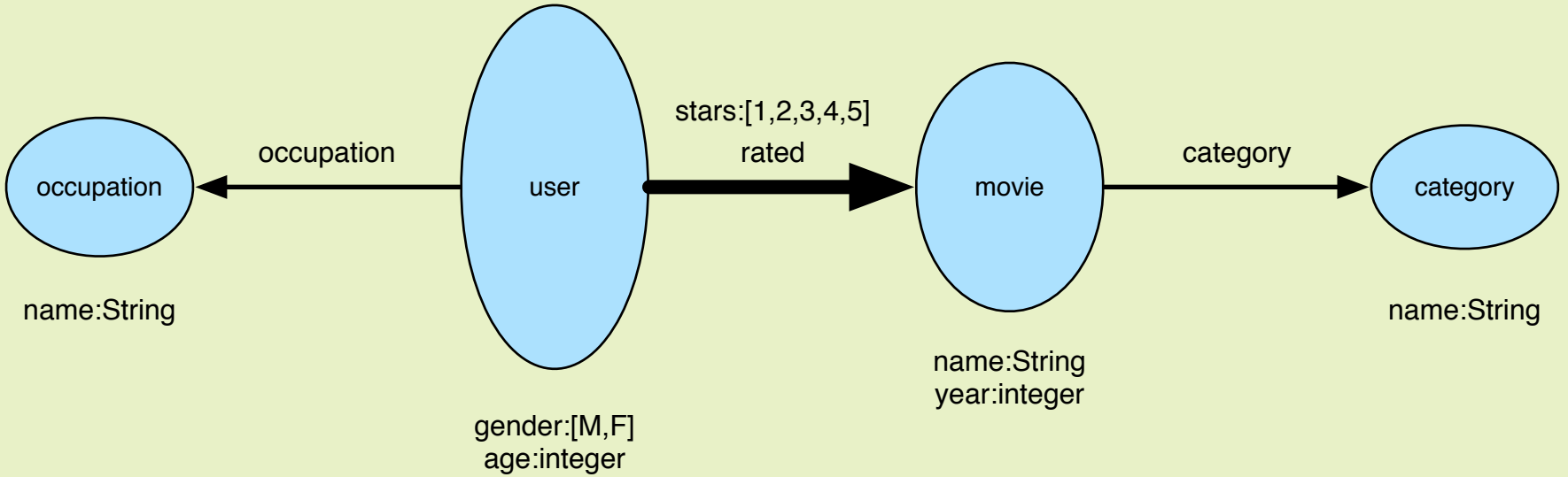
DATASTAX

2015 NOSQL NOW!

AUG 18-20 | SAN JOSE, CA

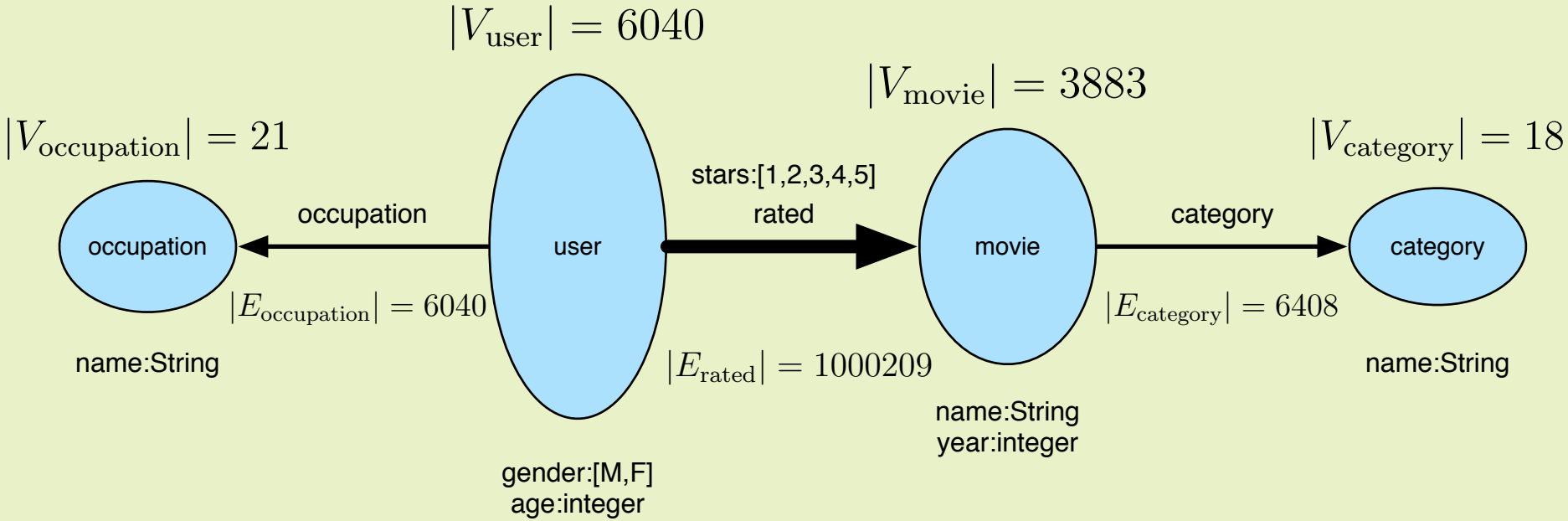
http://grouplens.org/datasets/movielens/

MovieLens Dataset



http://grouplens.org/datasets/movielens/

MovieLens Dataset



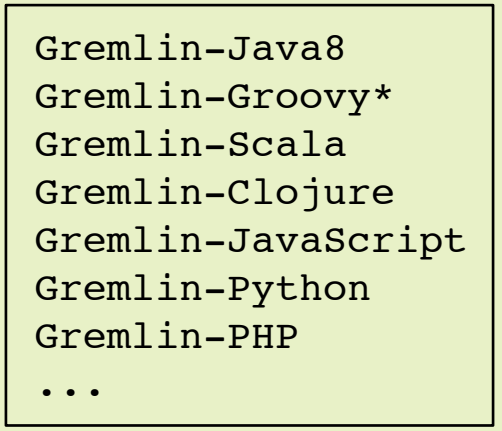
$$G = (V, E)$$

```
~/tinkerpop3$ bin/gremlin.sh
```

```
gremlin>
```

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin>
```



```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin>
```

"Create a new TinkerGraph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin>
```

$$G = (V = \emptyset, E = \emptyset)$$

G	The graph is a set of vertices and edges
V	The set of vertices in the graph
E	The set of edges in the graph
$\emptyset$	The empty set -- no elements

"Create a new TinkerGraph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin>
```

$$G = (V = \emptyset, E = \emptyset)$$

```
TitanGraph.open(…)
Neo4jGraph.open(…)
OrientGraph.open(…)
SqlgGraph.open(…)
HadoopGraph.open(…)
  GiraphGraphComputer
  SparkGraphComputer
ElasticGraph.open(…)
...
```

"Create a new TinkerGraph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin>
```

$$G = (V \neq \emptyset, E \subseteq (V \times V))$$

$A \subseteq B$ Set A is a subset of (or equal to) set B

$(V \times V)$ The set of all ordered pairs of vertices (directed binary edges)

"Load the MovieLens dataset into the newly created TinkerGraph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin>
```

"Create a graph traversal source for spawning graph traversals over the MovieLens graph."

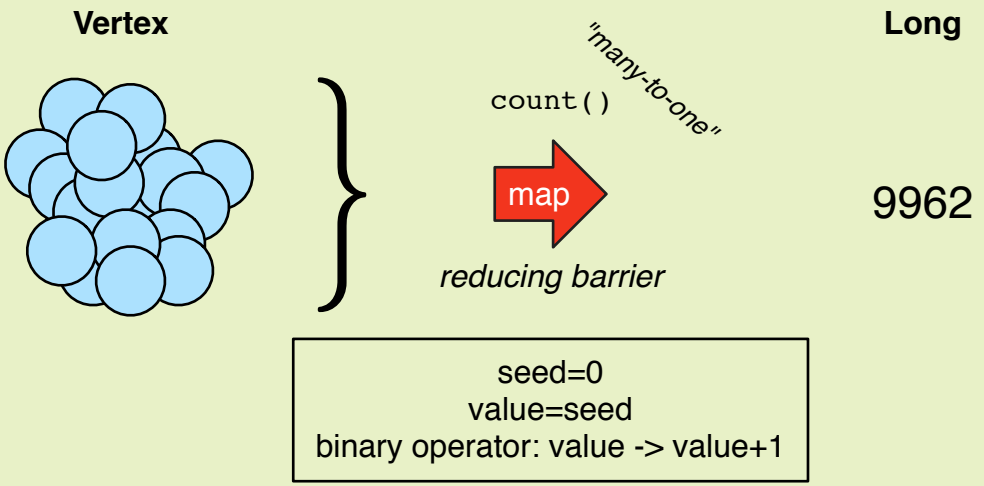
```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin>
```

$$|V| = 9962$$

"Count the number of vertices in the graph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin>
```

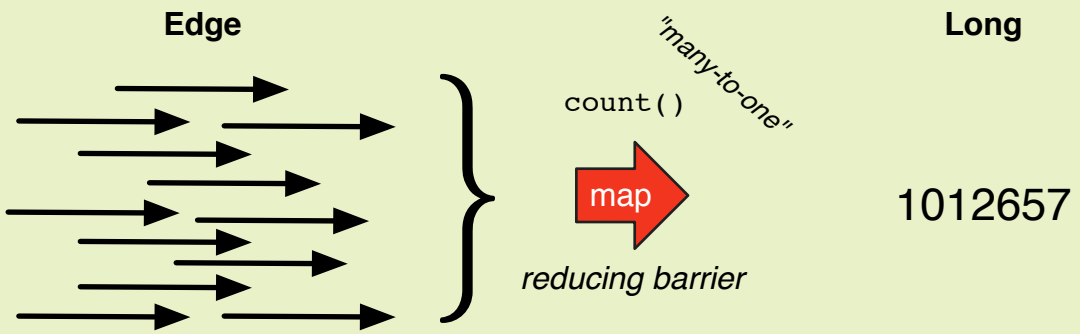
$$|V| = 9962$$



"Count the number of vertices in the graph."

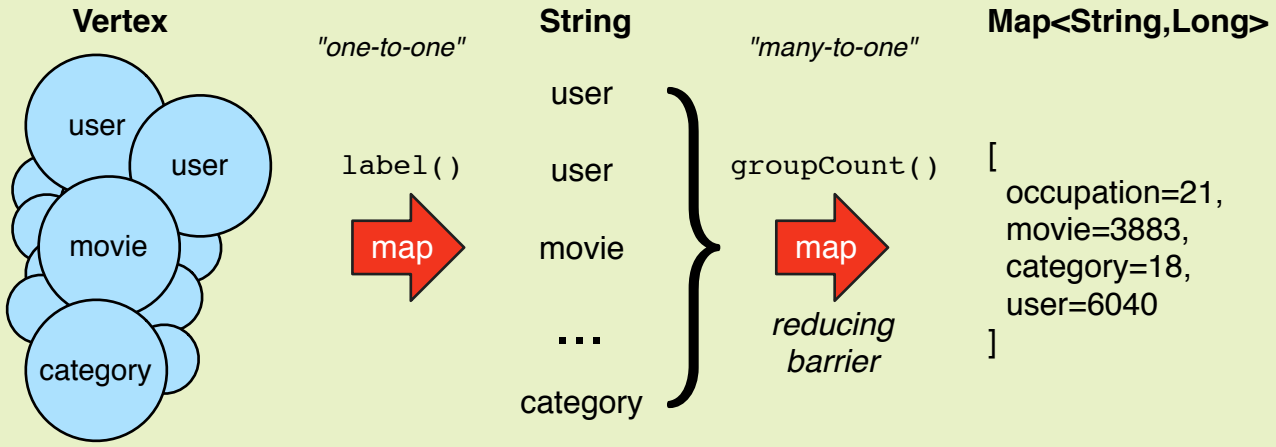
```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin> g.E().count()
==>1012657
gremlin>
```

$$|E| = 1012657$$



"Count the number of edges in the graph."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin> g.E().count()
==>1012657
gremlin> g.V().label().groupCount()
==>[occupation:21, movie:3883, category:18, user:6040]
gremlin>
```



"For each vertex in the graph, emit its label, then group and count each distinct label."

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin> g.E().count()
==>1012657
gremlin> g.V().label().groupCount()
==>[occupation:21, movie:3883, category:18, user:6040]
gremlin> g.E().hasLabel('rated').values('stars').mean()
==>3.581564453029317
gremlin>
```

"For each rated-edge in the graph, emit its stars property value and compute the average value."

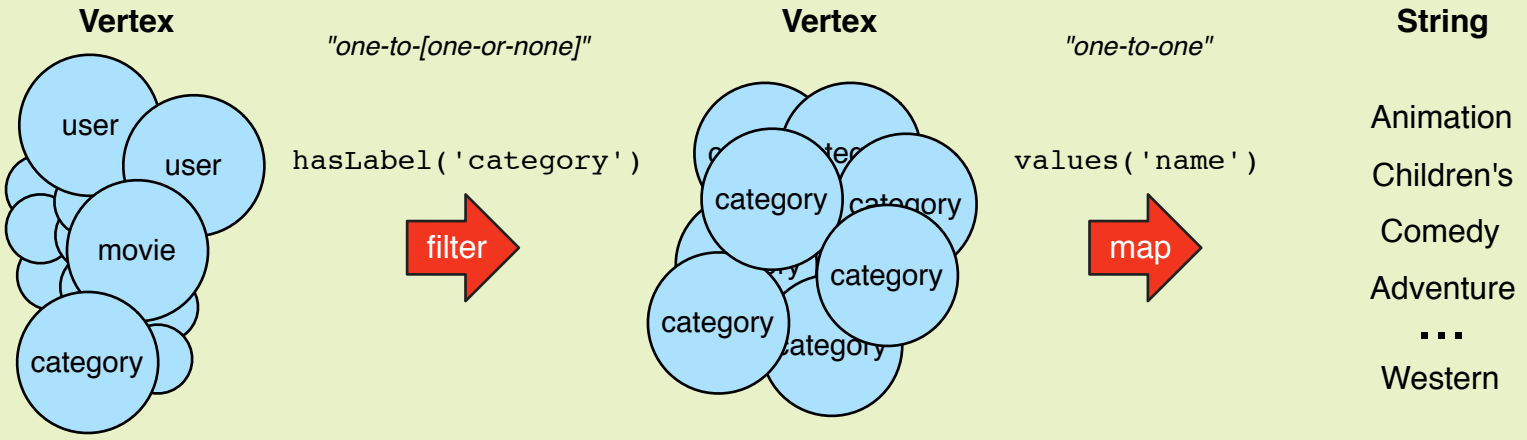
```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin> g.E().count()
==>1012657
gremlin> g.V().label().groupCount()
==>[occupation:21, movie:3883, category:18, user:6040]
gremlin> g.E().hasLabel('rated').values('stars').mean()
==>3.581564453029317
gremlin> g.V().hasLabel('user').map(outE('rated').count()).max()
==>2314
gremlin>
```

"What is the maximum number of movies a single user rated?"

```
~/tinkerpop3$ bin/gremlin.sh

         \,,,/
         (o o)
-----oOOo-(3)-oOOo-----
plugin activated: tinkerpop.server
plugin activated: tinkerpop.utilities
plugin activated: tinkerpop.tinkergraph
gremlin> graph = TinkerGraph.open()
==>tinkergraph[vertices:0 edges:0]
gremlin> graph.io(gryo()).readGraph('/tmp/movie-lens.kryo')
==>null
gremlin> g = graph.traversal()
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> g.V().count()
==>9962
gremlin> g.E().count()
==>1012657
gremlin> g.V().label().groupCount()
==>[occupation:21, movie:3883, category:18, user:6040]
gremlin> g.E().hasLabel('rated').values('stars').mean()
==>3.581564453029317
gremlin> g.V().hasLabel('user').map(outE('rated').count()).max()
==>2314
gremlin> g.V().hasLabel('movie').values('year').min()
==>1919
gremlin>
```

"What year was the oldest movie made?"

```
gremlin> g.V().hasLabel('category').values('name')
==>Animation
==>Children's
==>Comedy
==>Adventure
==>Fantasy
==>Romance
==>Drama
==>Action
==>Crime
==>Thriller
==>Horror
==>Sci-Fi
==>Documentary
==>War
==>Musical
==>Mystery
==>Film-Noir
==>Western
```



"For each vertex that is labeled 'category,' emit the name property value of that vertex."

```
gremlin> g.V().hasLabel('category').as('a','b').
           select('a','b').
             by('name').
             by(inE('category').count())
```

"For each category vertex, emit a map of its name and the number of movies it represents."

```
gremlin> g.V().hasLabel('category').as('a','b').
           select('a','b').
             by('name').
             by(inE('category').count())
```

$$V : \mathbb{G} \to V^*$$

$$\text{hasLabel}_{\text{category}} : V^* \to V^*$$

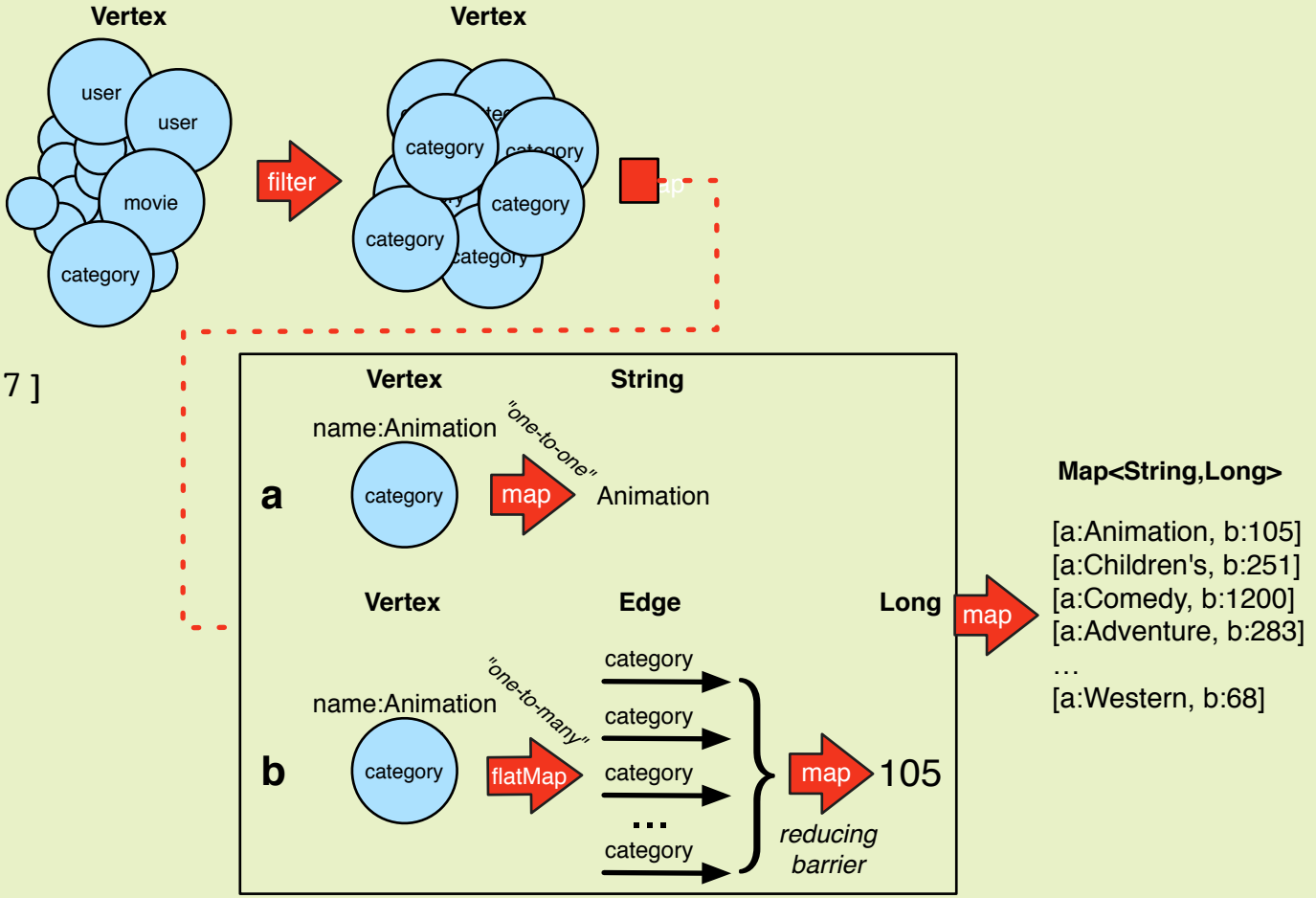
$$\text{as}_{\text{a,b}} : V^* \to (V \times V)^*$$

$$\text{select}_{\text{a,b}} : (V \times V)^* \to \begin{bmatrix} \text{a} & \text{values}_{\text{name}} : V^* \to \mathbb{S} \\ \text{b} & (\text{inE}_{\text{category}} : V^* \to E^*) \circ (\text{count} : E^* \to \mathbb{N}) \end{bmatrix} \to (\mathbb{S} \times \mathbb{N})^*$$

$f : A \to B$	The function f maps values of type A to values of type B
$\mathbb{G}$	The set of all graphs
A^*	A stream of values of type A
$(A \times B)$	The set of all pairs of values from A and B (cross product)
$\mathbb{N}$	The set of all natural numbers (1, 2, 3, 4, …)
$\mathbb{S}$	The set of all strings (a, b, aa, ab, bb, …) typically denoted Σ^*

"For each category vertex, emit a map of its name and the number of movies it represents."

```
gremlin> g.V().hasLabel('category').as('a','b').
           select('a','b').
             by('name').
             by(inE('category').count())
==>[a:Animation, b:105]
==>[a:Children's, b:251]
==>[a:Comedy, b:1200]
==>[a:Adventure, b:283]
==>[a:Fantasy, b:68]
==>[a:Romance, b:471]
==>[a:Drama, b:1603]
==>[a:Action, b:503]
==>[a:Crime, b:211]
==>[a:Thriller, b:492]
==>[a:Horror, b:343]
==>[a:Sci-Fi, b:276]
==>[a:Documentary, b:127]
==>[a:War, b:143]
==>[a:Musical, b:114]
==>[a:Mystery, b:106]
==>[a:Film-Noir, b:44]
==>[a:Western, b:68]
```

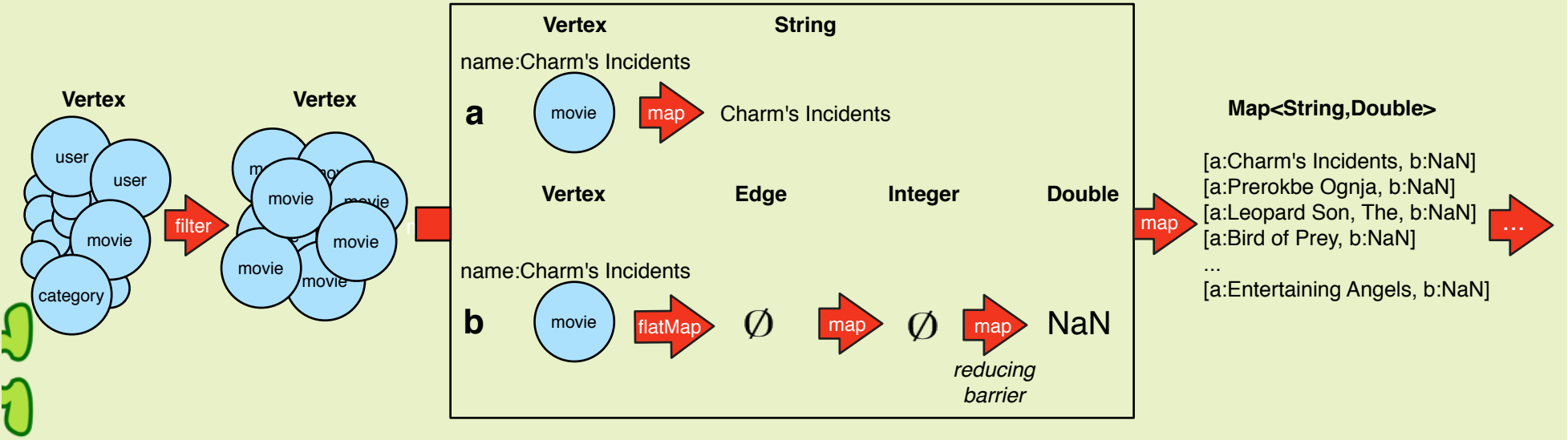


"For each category vertex, emit a map of its name and the number of movies it represents."

```
gremlin> g.V().hasLabel('movie').as('a','b').
           select('a','b').
             by('name').
             by(inE('rated').values('stars').mean()).
           order().by(select('b'),decr).
           limit(10)
```

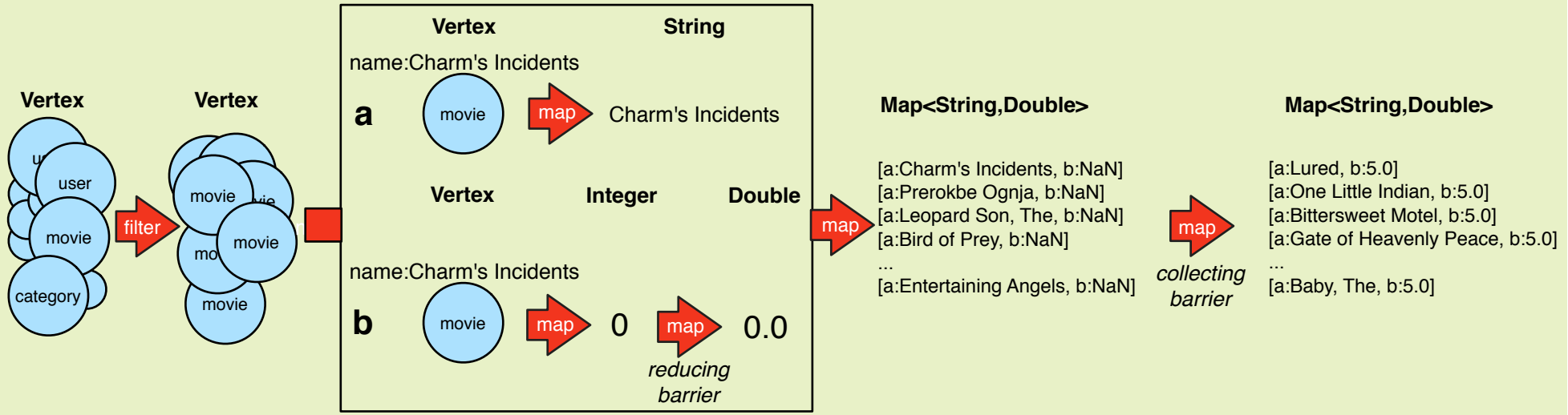
"For each movie, emit a map of its name and average rating.
Sort the maps in decreasing order by their average rating. Emit the first 10 maps (i.e. top 10)."

```
gremlin> g.V().hasLabel('movie').as('a','b').
           select('a','b').
             by('name').
             by(inE('rated').values('stars').mean()).
           order().by(select('b'),decr).
           limit(10)
==>[a:Charm's Incidents, b:NaN]
==>[a:Prerokbe Ognja, b:NaN]
==>[a:Leopard Son, The, b:NaN]
==>[a:Bird of Prey, b:NaN]
==>[a:Plutonium Circus, b:NaN]
==>[a:Hustler White, b:NaN]
==>[a:Curtis's Charm, b:NaN]
==>[a:Three Lives and Only One Death, b:NaN]
==>[a:Hoogste tijd, b:NaN]
==>[a:Entertaining Angels: The Dorothy Day Story, b:NaN]
```



"For each movie, emit a map of its name and average rating.
Sort the maps in decreasing order by their average rating. Emit the first 10 maps (i.e. top 10)."

```
gremlin> g.V().hasLabel('movie').as('a','b').
           select('a','b').
             by('name').
             by(coalesce(
                  inE('rated').values('stars'),
                  constant(0)).mean()).
           order().by(select('b'),decr).
           limit(10)
==>[a:Lured, b:5.0]
==>[a:One Little Indian, b:5.0]
==>[a:Bittersweet Motel, b:5.0]
==>[a:Gate of Heavenly Peace, The, b:5.0]
==>[a:Follow the Bitch, b:5.0]
==>[a:Schlafes Bruder (Brother of Sleep), b:5.0]
==>[a:Ulysses (Ulisse), b:5.0]
==>[a:Song of Freedom, b:5.0]
==>[a:Smashing Time, b:5.0]
==>[a:Baby, The, b:5.0]
```

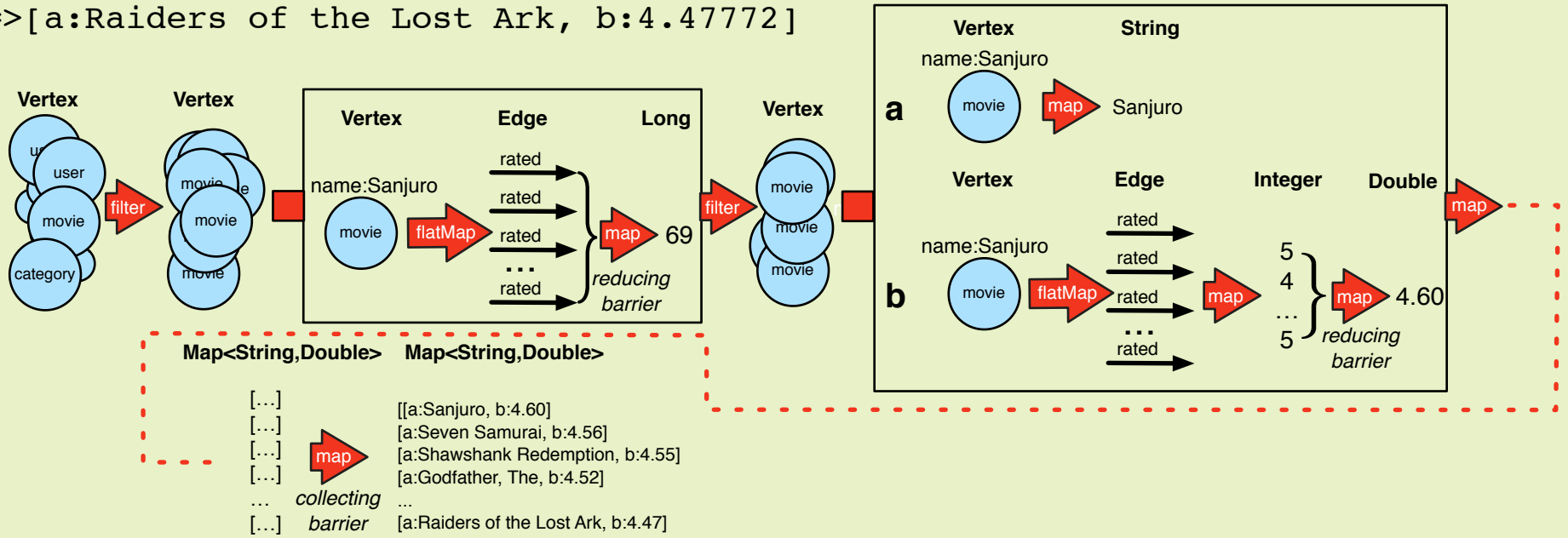


"For each movie, get its name and mean rating (or 0 if no ratings). Order by average rating and emit top 10."

```
gremlin> g.V().hasLabel('movie').as('a','b').
           where(inE('rated').count().is(gt(10))).
           select('a','b').
             by('name').
             by(inE('rated').values('stars').mean()).
           order().by(select('b'),decr).
           limit(10)
```

"For each movie with at least 11 ratings, emit a map of its name and average rating.
Sort the maps in decreasing order by their average rating. Emit the first 10 maps (i.e. top 10)."

```
gremlin> g.V().hasLabel('movie').as('a','b').
           where(inE('rated').count().is(gt(10))).
           select('a','b').
             by('name').
             by(inE('rated').values('stars').mean()).
           order().by(select('b'),decr).
           limit(10)
==>[a:Sanjuro, b:4.608695652173913]
==>[a:Seven Samurai (The Magnificent Seven), b:4.560509554140127]
==>[a:Shawshank Redemption, The, b:4.554557700942973]
==>[a:Godfather, The, b:4.524966261808367]
==>[a:Close Shave, A, b:4.52054794520548]
==>[a:Usual Suspects, The, b:4.517106001121705]
==>[a:Schindler's List, b:4.510416666666667]
==>[a:Wrong Trousers, The, b:4.507936507936508]
==>[a:Sunset Blvd. (a.k.a. Sunset Boulevard), b:4.491489361702127]
==>[a:Raiders of the Lost Ark, b:4.47772]
```



"For each movie with at least 11 ratings, emit a map of its name and average rating. Sort the maps in decreasing order by their average rating. Emit the first 10 maps (i.e. top 10)."

```
gremlin> g.V().hasLabel('movie').
           where(inE('rated').count().is(gt(10))).
           toString()
==>[GraphStep([],vertex), HasStep([~label.eq(movie)]),
    TraversalFilterStep([
      VertexStep(IN,[rated],edge),
      CountGlobalStep,
      IsStep(gt(10))])]
```

$$V : \mathbb{G} \to V^*$$

$$\text{hasLabel}_{\text{movie}} : V^* \to V^*$$

$$\text{where} : V^* \to \begin{array}{l} \text{inE}_{\text{rated}} : V^* \to E^* \\ \text{count} : E^* \to \mathbb{N} \\ \text{is}_{\text{gt}(10)} : \mathbb{N} \to (\mathbb{N} \cup \emptyset) \end{array} \to V^*$$

"true or false"

"What is the execution plan for the traversal prior to compiler optimizations being applied?"

```
gremlin> g.V().hasLabel('movie').
           where(inE('rated').count().is(gt(10))).
           iterate().toString()
==>[TinkerGraphStep(vertex,[~label.eq(movie)]),
    TraversalFilterStep([
      VertexStep(IN,[rated],edge),
      RangeGlobalStep(0,11),
      CountGlobalStep,
      IsStep(gt(10))])]
```

$$\mathrm{V}_{\mathrm{label=movie}} : \mathbb{G} \to V^*$$

* **TinkerGraphStragegy**: Access vendor-specific vertex partition by label.

$$\mathrm{where} : V^* \to \begin{array}{l} \mathrm{inE}_{\mathrm{rated}} : V^* \to E^* \\ \mathrm{limit}_{11} : E^* \to E^* \\ \quad \text{* } \textbf{RangeByIsCountStrategy}\text{: Only iterate 1 more than required count.} \\ \mathrm{count} : E^* \to \mathbb{N} \\ \mathrm{is}_{\mathrm{gt}(10)} : \mathbb{N} \to (\mathbb{N} \cup \emptyset) \end{array} \to V^*$$

"true or false"

"What is the execution plan for the traversal after compiler optimizations have been applied?"

```
gremlin> g.getStrategies()
==>ConjunctionStrategy
      a.and().b => and(a,b)
      a.or().b => or(a,b)
      a.or().b.and().c => or(a,and(b,c))
      a.and().b.or().c => or(and(a,b),c)
==>IncidentToAdjacentStrategy
      a.outE().inV().b => a.out().b
==>AdjacentToIncidentStrategy
      a.in().count().b => a.inE().count().b
      a.where(out()).b => a.where(outE()).b
      a.and(in(),out()).b => a.and(inE(),outE()).b
==>IdentityRemovalStrategy
      a.identity().b => a.b
==>FilterRankingStrategy
      a.order().dedup().b => a.dedup().order().b
      a.and(c,d).has().b => a.has().and(c,d).b
      a.simplePath().where().b => b.where().simplePath().a
==>MatchPredicateStrategy
      a.match(c,d).where(e).b => a.match(c,d,e)
      a.match(has(),c,d).b => a.has().match(c,d).b
==>RangeByIsCountStrategy
      a.count().is(0) => a.limit(1).count().is(0)
==>TinkerGraphStepStrategy
      V.has().has().b => V[has,has].b
==>ProfileStrategy
      a.b.c.profile() => a.profile().b.profile().c.profile()
==>ComputerVerificationStrategy
      a.order.b => IllegalStateException
      a.local(out().out()).b => IllegalStateException
```

"What compilation strategies are associated with the graph traversal source?"

```
gremlin> g.V().has('movie','name','Die Hard').
           inE('rated').values('stars').mean()
==>4.121848739495798
```

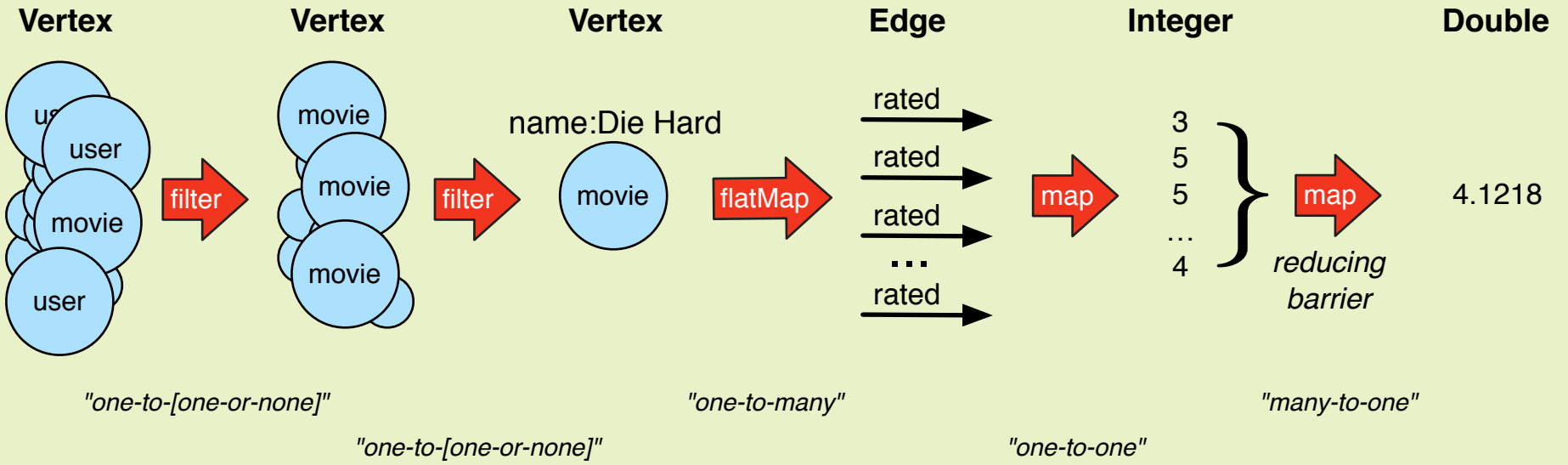
"What is Die Hard's average rating?"

```
gremlin> g.V().has('movie','name','Die Hard').
           inE('rated').values('stars').mean()
==>4.121848739495798
```

$$V : \mathbb{G} \rightarrow V^*$$

$$\text{has}_{\text{name=Die Hard}} : V^* \rightarrow V^* \qquad \text{values}_{\text{stars}} : E^* \rightarrow \mathbb{N}^*$$

$$\text{hasLabel}_{\text{movie}} : V^* \rightarrow V^* \qquad \text{inE}_{\text{rated}} : V^* \rightarrow E^* \qquad \text{mean} : \mathbb{N}^* \rightarrow \mathbb{R}$$

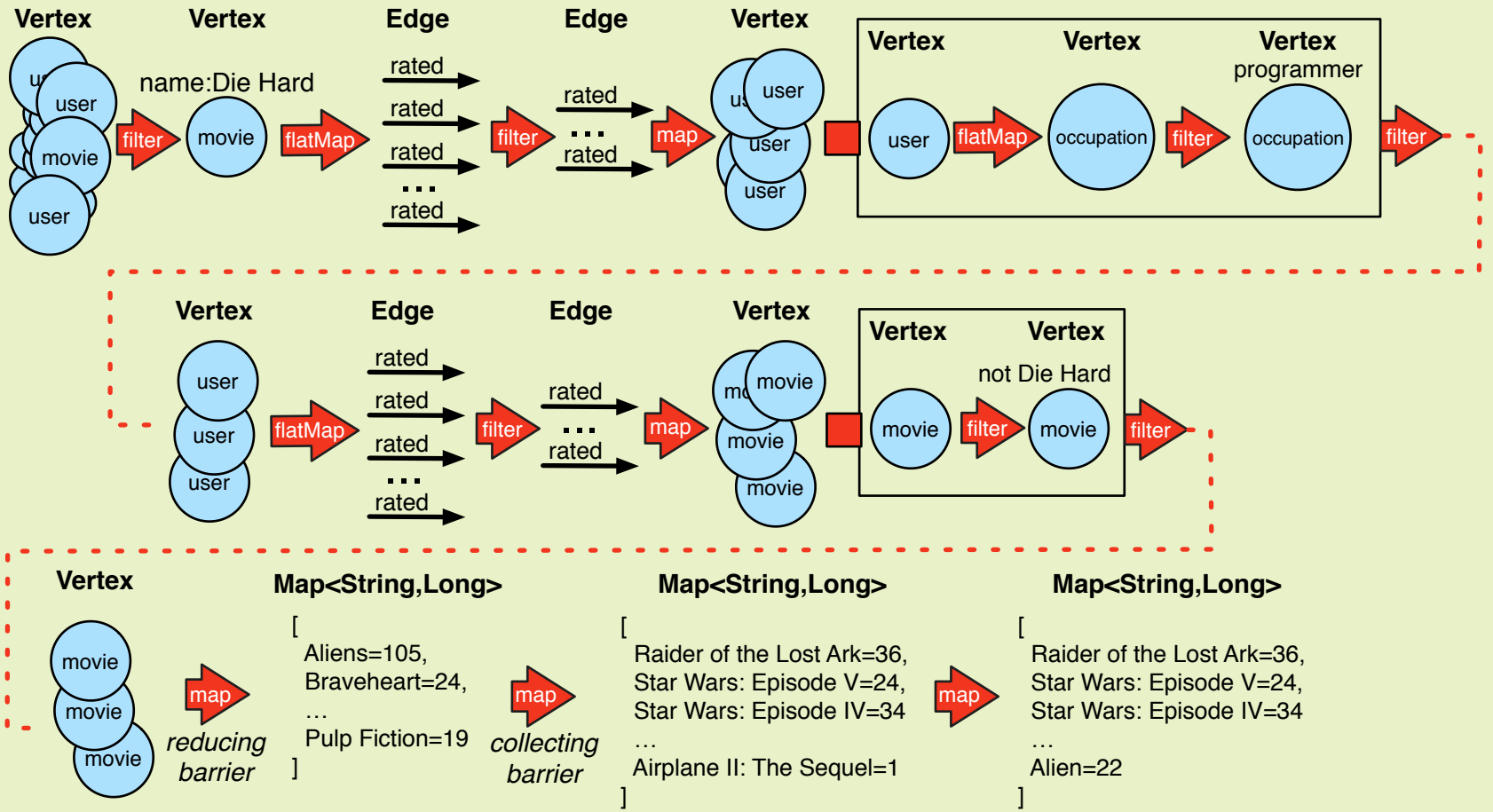


"What is Die Hard's average rating?"

```
gremlin> g.V().has('movie','name','Die Hard').as('a').
           inE('rated').has('stars',5).outV().
             where(out('occupation').has('name','programmer')).
           outE('rated').has('stars',5).inV().
             where(neq('a')).
           groupCount().by('name').
             order(local).by(valueDecr).
             limit(local,10).
             unfold()                        // so its not printed on a single line
```

"Which programmers like Die Hard and what other movies do they like?
Group and count the movies by their name. Sort the group count map in decreasing order by the count.
Clip the map to the top 10 entries and stream out the map's entries (for display purposes)."

```
gremlin> g.V().has('movie','name','Die Hard').as('a').
           inE('rated').has('stars',5).outV().
             where(out('occupation').has('name','programmer')).
           outE('rated').has('stars',5).inV().
             where(neq('a')).
           groupCount().by('name').
             order(local).by(valueDecr).
             limit(local,10).
             unfold()                          // so its not printed on a single line
```



"Which programmers like Die Hard and what other movies do they like?
Group and count the movies by their name. Sort the group count map in decreasing order by the count.
Clip the map to the top 10 entries and stream out the map's entries (for display purposes)."

```
gremlin> g.V().has('movie','name','Die Hard').as('a').
           inE('rated').has('stars',5).outV().
             where(out('occupation').has('name','programmer')).
           outE('rated').has('stars',5).inV().
             where(neq('a')).
           groupCount().by('name').
             order(local).by(valueDecr).
             limit(local,10).
             unfold()                      // so its not printed on a single line
==>Raiders of the Lost Ark=36
==>Star Wars: Episode V - The Empire Strikes Back=36
==>Star Wars: Episode IV - A New Hope=34
==>Matrix, The=32
==>Terminator, The=29
==>Star Wars: Episode VI - Return of the Jedi=26
==>Sixth Sense, The=26
==>Braveheart=24
==>Aliens=23
==>Alien=22
gremlin>
```

"Which programmers like Die Hard and what other movies do they like?
Group and count the movies by their name. Sort the group count map in decreasing order by the count.
Clip the map to the top 10 entries and stream out the map's entries (for display purposes)."

```
gremlin> g.V().
           match(
             __.as('a').hasLabel('movie'),
             __.as('a').out('category').has('name','Action'),
             __.as('a').has('year',between(1980,1990)),
             __.as('a').inE('rated').as('b'),
             __.as('b').has('stars',5),
             __.as('b').outV().as('c'),
             __.as('c').out('occupation').has('name','programmer'),
             __.as('c').has('age',between(30,40))).
           select('a').groupCount().by('name').
           order(local).by(valueDecr).
           limit(local,10).
           unfold()                        // so its not printed on a single line
```

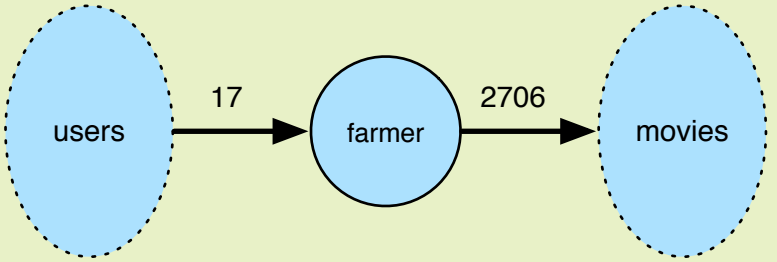
"What 80's action movies do 30-something programmers like?
Group count the movies by their name and sort the group count map in decreasing order by value.
Clip the map to the top 10 and emit the map entries."

```
gremlin> g.V().
           match(
             __.as('a').hasLabel('movie'),
             __.as('a').out('category').has('name','Action'),
             __.as('a').has('year',between(1980,1990)),
             __.as('a').inE('rated').as('b'),
             __.as('b').has('stars',5),
             __.as('b').outV().as('c'),
             __.as('c').out('occupation').has('name','programmer'),
             __.as('c').has('age',between(30,40))).
           select('a').groupCount().by('name').
           order(local).by(valueDecr).
           limit(local,10).
           unfold()                          // so its not printed on a single line
==>Raiders of the Lost Ark=26
==>Star Wars: Episode V - The Empire Strikes Back=26
==>Terminator, The=23
==>Star Wars: Episode VI - Return of the Jedi=22
==>Princess Bride, The=19
==>Aliens=18
==>Boat, The (Das Boot)=11
==>Indiana Jones and the Last Crusade=11
==>Star Trek: The Wrath of Khan=10
==>Abyss, The=9
gremlin>
```

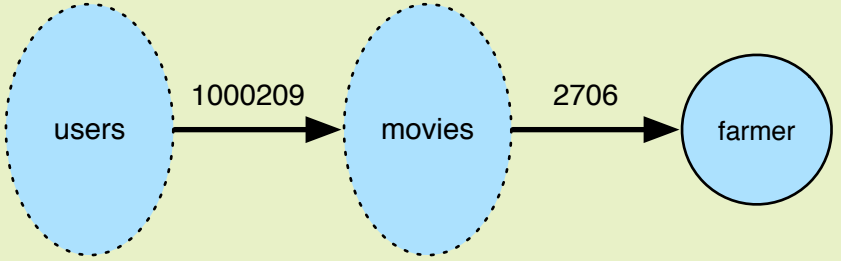
"What 80's action movies do 30-something programmers like?
Group count the movies by their name and sort the group count map in decreasing order by value.
Clip the map to the top 10 and emit the map entries."

```
MatchStep
  GraphTraversal.match(Traversal... traversalPatterns)

     x.match(
        a...b
        a...c
        c...
        or(
          a...c
          a...b
        )
        c.repeat(...).b
        not(c...a)
        b...count().e
        c...count().e
     ).dedup(a,b).y

a,b,c,e     : once a variable is set, it must hold equal for all patterns
c...        : "predicate patterns" simply check for the existence of a result
or()/and()  : nested conjunctive patterns supported
repeat(...) : recursive patterns supported
not(...)    : not'ing of patterns supported
count()     : barrier patterns supported
dedup(a,b)  : internal de-duplication of variable values supported
x.match().y : possible to go from imperative to declarative, etc.

Plug and Play MatchAlgorithms
  GreedyMatchAlgorithm :
     try each pattern in the order provided by the user
  CountMatchAlgorithm :
     continually re-sort patterns by the cardinality of their set reductions
```

```
// CountMatchAlgorithm (default)
gremlin> clockWithResult(50){
           g.V().match(
                  __.as('a').out('rated').as('b'),
                  __.as('a').out('occupation').has('name','farmer')).
                 select('a','b').count().next()}
==>66.31955294  // time in milliseconds
==>2706         // number of results
```




```
// GreedyMatchAlgorithm
gremlin> g = graph.traversal(GraphTraversalSource.build().
               with(MatchAlgorithmStrategy.build().
                    algorithm(MatchStep.GreedyMatchAlgorithm).create()))
==>graphtraversalsource[tinkergraph[vertices:9962 edges:1012657], standard]
gremlin> clockWithResult(50){
           g.V().match(
                  __.as('a').out('rated').as('b'),
                  __.as('a').out('occupation').has('name','farmer')).
                 select('a','b').count().next()}
==>1902.6290871599997  // time in milliseconds
==>2706                // number of results
```



"Which movies did each farmer rate? -- benchmark CountMatchAlgorithm vs. GreedyMatchAlgorithm."

```
gremlin> g.V().hasLabel('movie').
           where(inE('rated').count().is(gt(10))).
           group().
             by{((int)(it.value('year') / 10)) * 10}.
             by().
             by(unfold().order().
                         by(inE('rated').values('stars').mean(),decr).
                         values('name').
                         limit(1)).
           order(local).by(keyIncr).
           unfold()                        // so its not printed on a single line
```

"What is the most liked movie in each decade?"

```
gremlin> g.V().hasLabel('movie').
           where(inE('rated').count().is(gt(10))).
           group().
             by{((int)(it.value('year') / 10)) * 10}.
             by().
             by(unfold().order().
                         by(inE('rated').values('stars').mean(),decr).
                         values('name').
                         limit(1)).
           order(local).by(keyIncr).
           unfold()                        // so its not printed on a single line
```

λ

Nearly every step that takes a traversal argument can also take a lambda.
It is recommended that users do not use lambdas as they are not subject to traversal strategy (i.e. compiler) optimization.
However, they are useful when no provided step yields the desired computation.

"What is the most liked movie in each decade?"

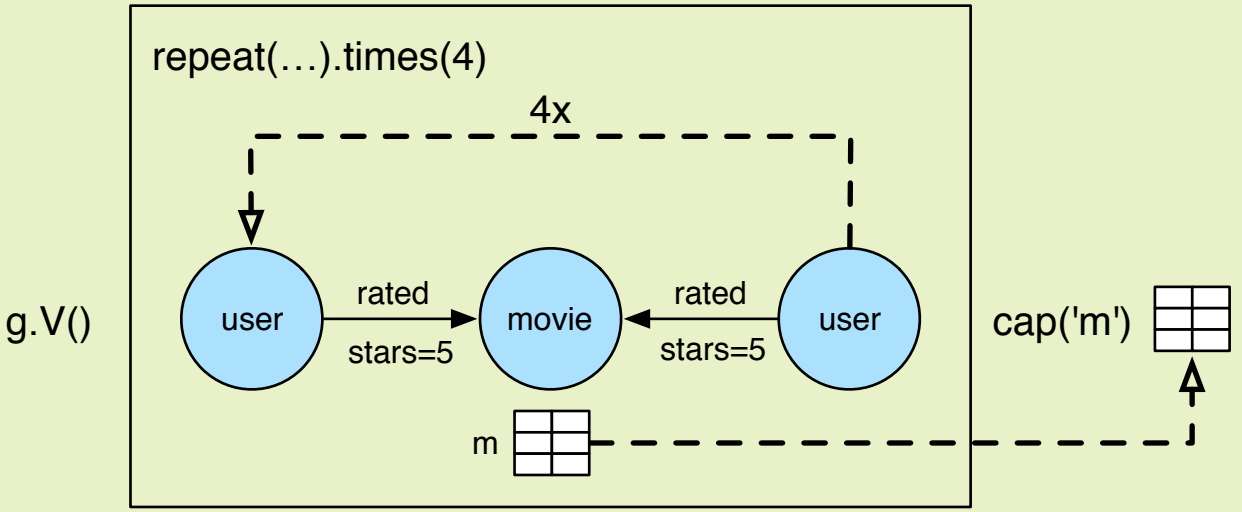
```
gremlin> g.V().hasLabel('movie').
           where(inE('rated').count().is(gt(10))).
           group().
             by{((int)(it.value('year') / 10)) * 10}.
             by().
             by(unfold().order().
                         by(inE('rated').values('stars').mean(),decr).
                         values('name').
                         limit(1)).
           order(local).by(keyIncr).
           unfold()                          // so its not printed on a single line
==>1910=Daddy Long Legs
==>1920=General, The
==>1930=City Lights
==>1940=Third Man, The
==>1950=Seven Samurai (The Magnificent Seven)
==>1960=Sanjuro
==>1970=Godfather, The
==>1980=Raiders of the Lost Ark
==>1990=Shawshank Redemption, The
==>2000=Almost Famous
gremlin>
```

"What is the most liked movie in each decade?"

```
gremlin> graph = HadoopGraph.open('conf/hadoop/movie-lens.properties')
==>hadoopgraph[gryoinputformat->gryooutputformat]
gremlin> g = graph.traversal(computer(SparkGraphComputer))
==>graphtraversalsource
     [hadoopgraph[gryoinputformat->gryooutputformat], sparkgraphcomputer]
gremlin>
```

"Which movies are most central in the implicit 5-stars graph?"

```
gremlin> graph = HadoopGraph.open('conf/hadoop/movie-lens.properties')
==>hadoopgraph[gryoinputformat->gryooutputformat]
gremlin> g = graph.traversal(computer(SparkGraphComputer))
==>graphtraversalsource
     [hadoopgraph[gryoinputformat->gryooutputformat], sparkgraphcomputer]
gremlin> g.V().repeat(outE('rated').has('stars', 5).inV().
                  groupCount('m').by('name').
                  inE('rated').has('stars', 5).outV()).
                times(4).cap('m')
```



"Which movies are most central in the implicit 5-stars graph?"

```
gremlin> graph = HadoopGraph.open('conf/hadoop/movie-lens.properties')
==>hadoopgraph[gryoinputformat->gryooutputformat]
gremlin> g = graph.traversal(computer(SparkGraphComputer))
==>graphtraversalsource
     [hadoopgraph[gryoinputformat->gryooutputformat], sparkgraphcomputer]
gremlin> g.V().repeat(outE('rated').has('stars', 5).inV().
                groupCount('m').by('name').
                inE('rated').has('stars', 5).outV()).
              times(4).cap('m')
==>Fantasia 2000=2676505178171564
==>Pale Rider=1369969000295362
==>Crucible, The=401712993698149
==>About Adam=37981148456999
==>Akira=3659939409345918
...
gremlin>
```

"Which movies are most central in the implicit 5-stars graph?"

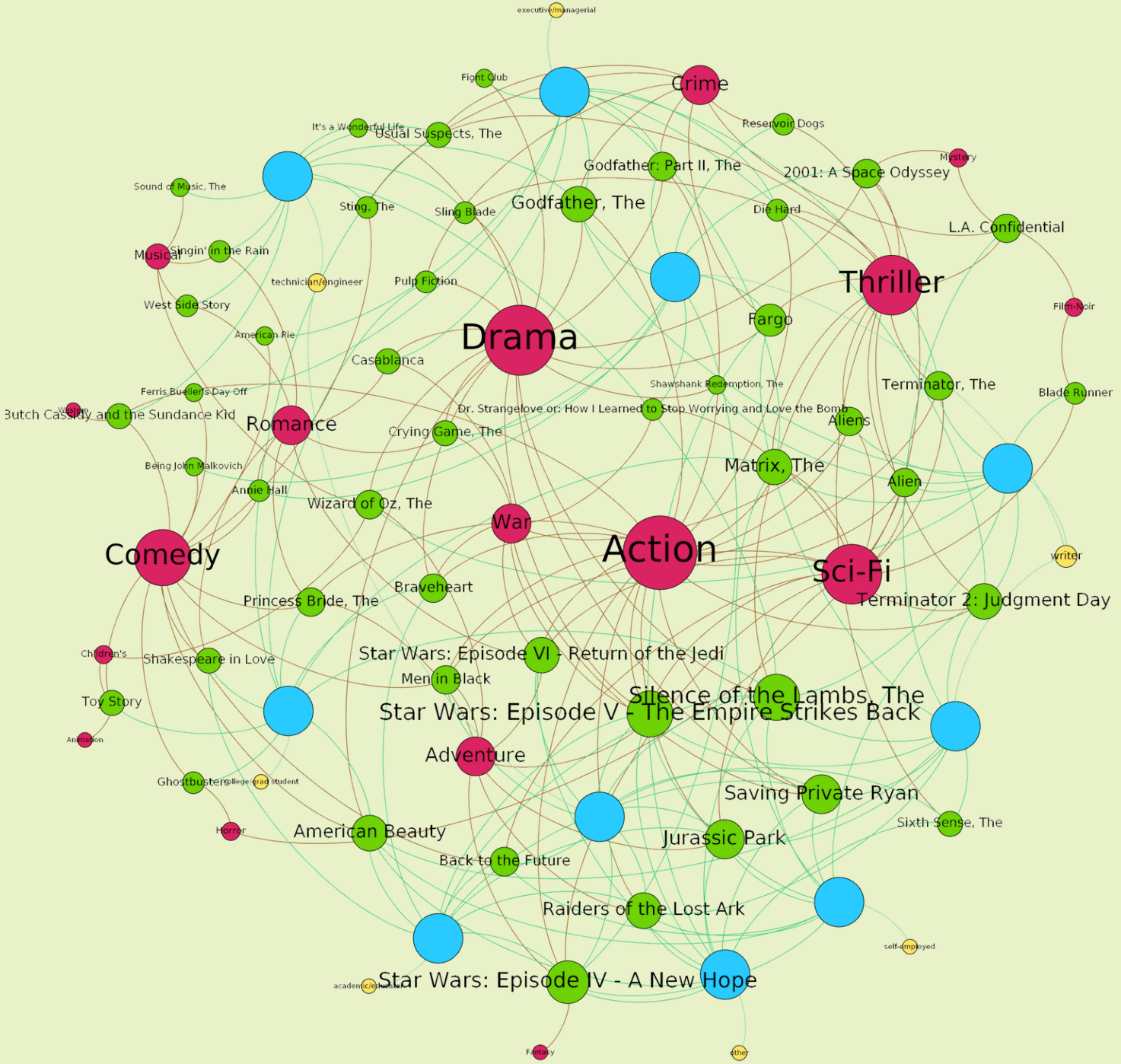
```
gremlin> graph = HadoopGraph.open('conf/hadoop/movie-lens.properties')
==>hadoopgraph[gryoinputformat->gryooutputformat]
gremlin> g = graph.traversal(computer(SparkGraphComputer))
==>graphtraversalsource
     [hadoopgraph[gryoinputformat->gryooutputformat], sparkgraphcomputer]
gremlin> g.V().repeat(outE('rated').has('stars', 5).inV().
                groupCount('m').by('name').
                inE('rated').has('stars', 5).outV()).
              times(4).cap('m')
==>Fantasia 2000=2676505178171564
==>Pale Rider=1369969000295362
==>Crucible, The=401712993698149
==>About Adam=37981148456999
==>Akira=3659939409345918
...
gremlin> hdfs.ls('output/m')
==>rw-r--r-- daniel supergroup 0 _SUCCESS
==>rw-r--r-- daniel supergroup 245314 part-r-00000
gremlin> hdfs.head('output/m', ObjectWritable).sort {-it.value}.take(10)
==>Star Wars: Episode IV - A New Hope  3540539435310 5332
==>American Beauty  3194322828202 0585
==>Raiders of the Lost Ark  3122477979323 8499
==>Star Wars: Episode V - The Empire Strikes Back  3043467711972 6223
==>Godfather, The  3025851852301 3057
==>Shawshank Redemption, The  2829771738790 1031
==>Schindler's List  2753933665419 9309
==>Silence of the Lambs, The  2673627637680 6173
==>Fargo  2653105031132 5270
==>Matrix, The  2639511823920 3191
```

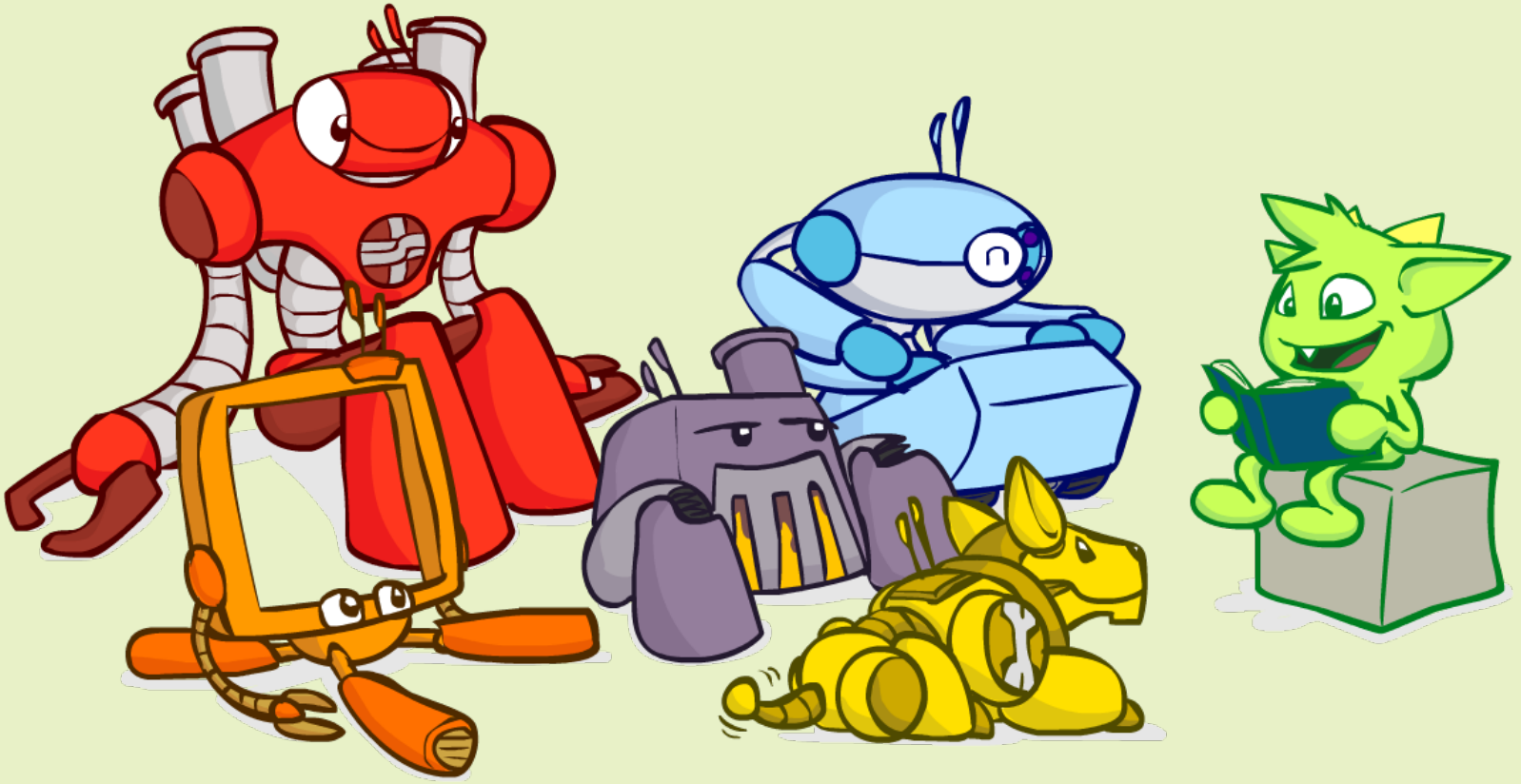
"Which movies are most central in the implicit 5-stars graph?"

```
gremlin> :plugin use tinkerpop.gephi
==>tinkerpop.gephi activated
gremlin> :remote connect tinkerpop.gephi
==>Connection to Gephi - http://localhost:8080/workspace0 with stepDelay:1000,
startRGBColor:[0.0, 1.0, 0.5], colorToFade:g, colorFadeRate:0.7, startSize:
20.0,sizeDecrementRate:0.33
gremlin>
```

```
gremlin> :plugin use tinkerpop.gephi
==>tinkerpop.gephi activated
gremlin> :remote connect tinkerpop.gephi
==>Connection to Gephi - http://localhost:8080/workspace0 with stepDelay:1000,
startRGBColor:[0.0, 1.0, 0.5], colorToFade:g, colorFadeRate:0.7, startSize:
20.0,sizeDecrementRate:0.33
gremlin> :> g.V().hasLabel('user').                                    "movie buffs"
             order().
               by(outE('rated').count(), decr).limit(10).as('a').
             local(outE('rated').order().
                 by('stars', decr).                    // first by stars
                 by(inV().inE('rated').count(), decr). // then by ratings
               limit(10)).
             subgraph('sg').inV().outE('category').
             subgraph('sg').select('a').outE('occupation').
             subgraph('sg').cap('sg').next()
==>tinkergraph[vertices:82 edges:233]
gremlin>
```

"Which users rated the most movies?
For each user, display their 10 favorite movies, the categories of those movies, and their occupation.


executive/managerial
Fight Club
Crime
It's a Wonderful Life
Usual Suspects, The
Reservoir Dogs
Mystery
Godfather: Part II, The
2001: A Space Odyssey
Sound of Music, The
Sting, The
Sling Blade
Godfather, The
Die Hard
L.A. Confidential
Musical
Singin' in the Rain
technician/engineer
Pulp Fiction
Thriller
West Side Story
Film-Noir
Fargo
American Pie
Drama
Casablanca
Shawshank Redemption, The
Terminator, The
Blade Runner
Ferris Bueller's Day Off
Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb
Butch Cassidy and the Sundance Kid
Romance
Crying Game, The
Aliens
Being John Malkovich
Matrix, The
Annie Hall
Alien
Wizard of Oz, The
War
Comedy
Action
Sci-Fi
writer
Braveheart
Princess Bride, The
Terminator 2: Judgment Day
Children's
Shakespeare in Love
Star Wars: Episode VI - Return of the Jedi
Men in Black
Toy Story
Silence of the Lambs, The
Star Wars: Episode V - The Empire Strikes Back
Animation
Adventure
Ghostbusters
college grad student
Saving Private Ryan
Horror
American Beauty
Jurassic Park
Sixth Sense, The
Back to the Future
Raiders of the Lost Ark
self-employed
academic/educator
Star Wars: Episode IV - A New Hope
Fantasy



Thanks for listening…