Differentiated access control to graph data

Notebook output

Marc de Lignie

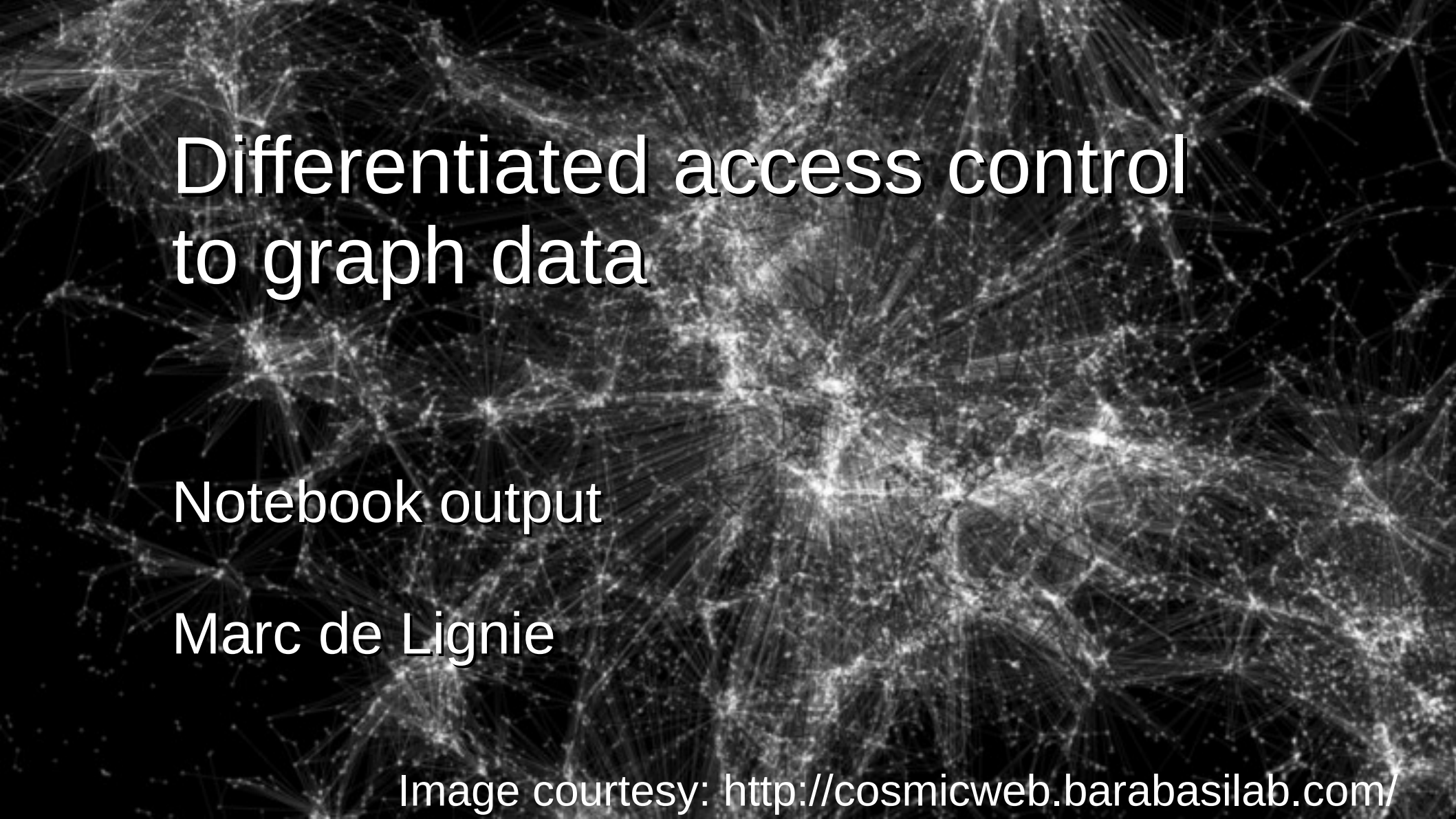


Image courtesy: http://cosmicweb.barabasilab.com/

Demo notebook for FOSDEM2019 graph devroom

In [1]:

```
/* Tested with beakerx-1.3.0 on python-3.6.7 */

%classpath add jar 'target/fosdem2019-0.0.1-SNAPSHOT.jar'
%classpath add jar 'gremlinjars/lib/*'

import nl.vtslab.fosdem2019.traversal.AuthorizedTraversalSource
import static org.apache.tinkerpop.gremlin.structure.
  VertexProperty.Cardinality.set;
import org.apache.tinkerpop.gremlin.tinkergraph.structure.
  TinkerGraph

class DemoGraph {
  static def createGraph() {
    def graph = TinkerGraph.open();
    def g = graph.traversal();
    g.addV("Person").property("name", "p0").
      property("authz", "biz;3").
      property(set, "authz", "fb;2").next();
    g.addV("Person").property("name", "p1").
      property("authz", "fin;3").next();
    g.addV("Person").property("name", "p2").
      property("authz", "fin;2").next();
    g.addV("Event").property("name", "v1").
      property("authz", "biz;3").next();
    g.addV("Event").property("name", "v2").
      property("authz", "fin;2").next();
```

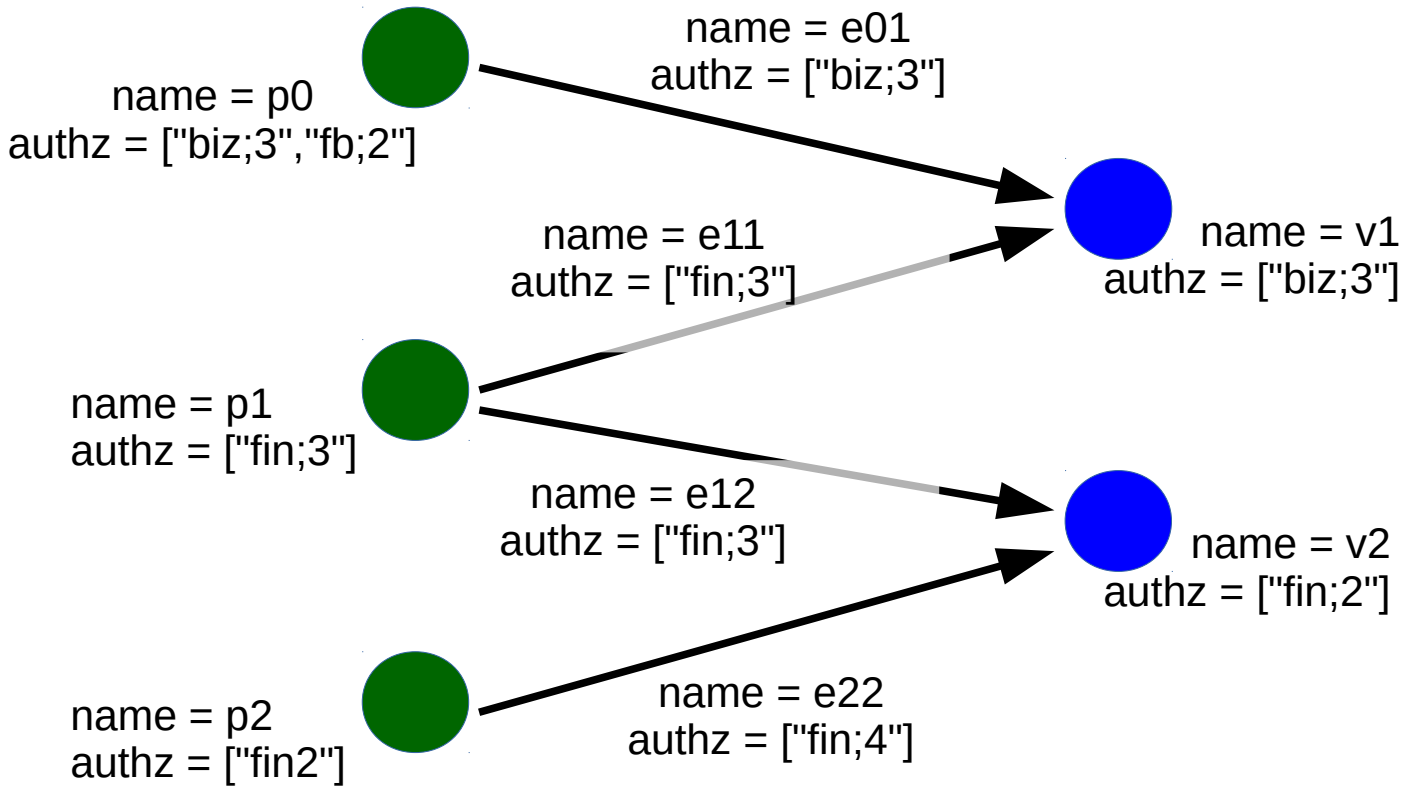
```
    g.V().has("name", "v1").as("a").
        V().has("name", "p0").addE("Visits").to("a").
          property("name", "e01").property("authz", "biz;3").
        V().has("name", "p1").addE("Visits").to("a").
          property("name", "e11").
          property("authz", "fin;3").next();
    g.V().has("name", "v2").as("a").
        V().has("name", "p1").addE("Visits").to("a").
          property("name", "e12").property("authz", "fin;3").
        V().has("name", "p2").addE("Visits").to("a").
          property("name", "e22").
          property("authz", "fin;4").next();
    return graph;
  }
}

graph = DemoGraph.createGraph();
g = graph.traversal();
// Every withAuthorization() call needs
// a new AuthorizedTraversalSource
ga = {x -> graph.traversal(AuthorizedTraversalSource.class)}
"graph and traversal sources available"
```

▸ Added jar: fosdem2019-0.0.1-SNAPSHOT.jar

▸ Added jars: commons-lang3-3.3.1.jar, jcl-over-slf4j-1.7.21.jar, gremlin-core-3.2.9.jar, ...

Out[1]: graph and traversal sources available


name = p0
authz = ["biz;3","fb;2"]
name = e01
authz = ["biz;3"]
name = v1
authz = ["biz;3"]
name = e11
authz = ["fin;3"]
name = p1
authz = ["fin;3"]
name = e12
authz = ["fin;3"]
name = v2
authz = ["fin;2"]
name = p2
authz = ["fin2"]
name = e22
authz = ["fin;4"]

Normal operation

In [2]:
```
unrestricted = g.V().values("name").toList()
authorized = ga().withAuthorization(["biz;3"]).
    V().values("name").toList()    // <===
"unrestricted: " + unrestricted +
    "\nauthorized: " + authorized
```

Out[2]:
```
unrestricted: [p0, p1, p2, v1, v2]
authorized: [p0, v1]
```

In [3]:
```
unrestricted = g.V().has("name", "p1").
    out().values("name").toList()
authorized = ga().withAuthorization(["fin;2","fin;3"]).
    V().has("name", "p1").
    out().values("name").toList()      // <===
"unrestricted: " + unrestricted +
    "\nauthorized: " + authorized
```

Out[3]:
```
unrestricted: [v1, v2]
authorized: [v2]
```

Unauthorized use

In [4]:
```
try {
    ga().V().toList();      // <===
    fail("Unauthorized query should fail");
} catch (RuntimeException exception) {
    exception.getMessage();
}
```

Out[4]: Method withAuthorization() should be called first

In [5]:
```
try {
    ga().withAuthorization(["biz;3"]).
        withAuthorization(["biz;3", "fin;3"]).
        V().toList();      // <===
    fail("Query with second withAuthorization() " +
         "call should fail");
} catch (RuntimeException exception) {
    exception.getMessage();
}
```

Out[5]: Method withAuthorization() can only be called once

Trying to manipulate the internal userAuthorization variable

In [6]:
```
ga().withAuthorization(["biz;3"]).
    V().cap("userAuthorization").next()
```

Out[6]:
```
[biz;3]
```

In [7]:
```
authorized = ga().withAuthorization(["biz;3"]).
    withSideEffect("userAuthorization", ["biz;3", "fin;3"]).
    V().values("name").toList();      // <===
"" + authorized +
    " withSideEffect() to set userAuthorization is ignored"
// Calling anything, including withSideEffect(), before
// withAuthorization() results in an exception
```

Out[7]:
```
[p0, v1] withSideEffect() to set userAuthorization is ignored
```

In [8]:

```
try {
    ga().withAuthorization(["biz;3"]).V().
        as("x").inject("fin;3").store("userAuthorization").
        select("x").toList();      // <===
    fail("Faking userAuthorizations using " +
         "the store() step should fail");
} catch (UnsupportedOperationException exception) {
    "Cannot add to unmodifiable list: " + exception.getClass()
}
// Using aggregate() step instead of
// store() step gives same result
```

Out[8]:

```
Cannot add to unmodifiable list: class java.lang.UnsupportedOperation
Exception
```

In [9]:

```
try {
    authorizations = ["fin;2", "fin;4"];
    result = ga().withAuthorization(["fin;2"]).
        V().has("name", "p2").map({t ->
            t.sideEffects("userAuthorization", authorizations);
            return t.get();
            }).out().toList();      // <===
    fail("Accessing AuthorizedTraversal.map(Function) " +
         "should fail");
} catch (RuntimeException exception) {
    exception.getMessage()
}
// Apart from the map() step also the barrier(), branch(),
// flatMap(), filter(), emit(), sideEffect() and until()
// steps provide Traverser instances and need blocking
```

Out[9]:

```
Method not available for AuthorizedTraversal
```

Trying to access the graph instance

In [10]:
```
try {
    ga().withAuthorization(["biz;3"]).V().getGraph().
        get().traversal().V().toList();
    fail("Accessing DefaultAuthorizedTraversal.getGraph() " +
         " should fail");
} catch (RuntimeException exception) {
    exception.getMessage()
}
```

Out[10]:
```
Method not available for AuthorizedTraversal
```

In [11]:
```
testClass = Class.forName("DemoGraph");
testObject = testClass.getConstructor().newInstance();
vertices = testObject.createGraph().
    traversal().V().values("name").toList();
"Applications using AuthorizedTraversalSource need to set \n" +
    "the JVM SecurityManager: " + vertices
```

Out[11]:
```
Applications using AuthorizedTraversalSource need to set
the JVM SecurityManager: [p0, p1, p2, v1, v2]
```

Trying to access the graph via an anonymous traversal

In [12]:
```
import nl.vtslab.fosdem2019.traversal.__

result = ga().withAuthorization(["biz;3"]).V().
    map(__.V().fold()).unfold().dedup().values("name").toList()
"Anonymous __.V() inherits userAuthorization from parent: " +
    result
```

Out[12]:
```
Anonymous __.V() inherits userAuthorization from parent: [p0, v1]
```

In [13]:
```
import nl.vtslab.fosdem2019.traversal.__
result = ga().withAuthorization(["biz;3"]).V().has("name", "v1").
    map(__.inE().fold()).unfold().values("name").toList()
"Anonymous __.inE() inherits userAuthorization from parent: " +
    result
```

Out[13]:
```
Anonymous __.inE() inherits userAuthorization from parent: [e01]
```